High-Performance Scalable Base-4 Fast Fourier Transform Mapping

Dr. J. Greg Nash, Centar
(jgregnash@centar.net, www.centar.net)

A novel, scalable parallel FFT architecture mapping is described here that supports transform lengths which aren't powers of two or four, that provides low latency as well as high throughput, that can do both 1-D and 2-D discreet Fourier transforms (DFTs), that is ideally suited to today's complex FPGA architectures, that possesses all the regularity and design simplicity of systolic arrays and that is naturally suited to a parameterized HDL form. Its algorithmic underpinnings are based on an observation that with suitable permutations, the DFT coefficient matrix can be partitioned into regular blocks of smaller "base-4" matrices (equivalent to a decimation in time *and* frequency) [1]. From this new base-4 matrix DFT description we have derived a new latency and throughput optimal base-4 FFT architecture. It combines the performance of traditional radix-4 "pipelined FFTs" with the design and implementation simplicity of systolic arrays, and yet is versatile.

An N point DFT is defined by

$$Z[k] = \sum_{n=1}^{N} X[n]\, e^{-j(2\pi/N)(k-1)(n-1)} \quad k = 1,2\ldots N \quad \text{or} \quad Z = CX \tag{1}$$

where $X[n]$ are the time domain input values, $Z[k]$ are the frequency domain outputs and C is a coefficient matrix containing elements $W_N^{kn} = e^{-2j\pi(n-1)(k-1)/N}$. In order to transform C into the desired base-b ($b=4$) format it is necessary to find a permutation matrix P that reorders X and Z according to

$$\begin{aligned} X_b &= P\ [X_1\, X_2\, X_3\, X_4\, X_5 \ldots X_{N-3}\, X_{N-2}\, X_{N-1}\, X_N]^t \\ &= [X_1\, X_{1+N/4}\, X_{1+N/2}\, X_{1+3N/4}\, X_2 \ldots X_{N/4}\, X_{N/2}\, X_{3N/4}\, X_N]^t . \end{aligned} \tag{2}$$

and $Z_b = P\ Z$. With this value of P, C can be transformed into $C_b = PCP^t$, so that $Z_b = C_b X_b$. This transformation allows C_b to be written as an $(N/b)\times(N/b)$ array of bxb blocks, each block $C_b[i,j]$ specified by

$C_b[i,j] = W_M[i,j] * c_{D((j-1)\bmod(b))+1} C_{((i-1)\bmod(b))+1}$ where $c_{Di} = c_i^t I$ with c_i a b-element vector, C_i is a bxb matrix, each row being c_i^t, and $W_M[i,j]$ is an element in the $(N/b)\times(N/b)$ matrix,

$$W_M = \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots \\ 1 & W^1 & W^2 & W^3 & \cdots \\ 1 & W^2 & W^4 & W^6 & \cdots \\ 1 & W^3 & W^6 & W^9 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}. \tag{3}$$

With this block reformulation it is possible to factor (1) into

$$\begin{aligned} Y &= W_M \bullet C_{M1} X \\ Z &= C_{M2} Y^t \end{aligned} \tag{4}$$

where "●" in (4) corresponds to an element by element multiply [2]. In (4) C_{M1} and C_{M2} contain N/b^2 submatrices $C_B = [c_1 \mid c_2 \mid \ldots \mid c_b]^t$ with the form $C_{M1} = \left[C_B^t \mid C_B^t \mid \ldots\right]^t$ and $C_{M2} = [C_B \mid C_B \mid \ldots]$, and Z, X have been redefined as follows:

$$Z = \begin{bmatrix} Z_1 & & Z_{N/4} \\ Z_{1+N/4} & \cdots & Z_{N/2} \\ Z_{1+N/2} & & Z_{3N/4} \\ Z_{1+3N/4} & & Z_N \end{bmatrix}, X = \begin{bmatrix} X_1 & & X_{N/4} \\ X_{1+N/4} & \cdots & X_{N/2} \\ X_{1+N/2} & & X_{3N/4} \\ X_{1+3N/4} & & X_N \end{bmatrix}. \tag{5}$$

For base-4 designs (b=4), $c_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$, $c_2 = \begin{bmatrix} 1 \\ -j \\ -1 \\ j \end{bmatrix}$, $c_3 = \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \end{bmatrix}$, $c_4 = \begin{bmatrix} 1 \\ j \\ -1 \\ -j \end{bmatrix}$

Report Documentation Page	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE **20 AUG 2004**	2. REPORT TYPE **N/A**	3. DATES COVERED **-**
4. TITLE AND SUBTITLE **High-Performance Scalable Base-4 Fast Fourier Transform Mapping**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Centar**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)
12. DISTRIBUTION/AVAILABILITY STATEMENT **Approved for public release, distribution unlimited**		
13. SUPPLEMENTARY NOTES **See also ADM001694, HPEC-6-Vol 1 ESC-TR-2003-081; High Performance Embedded Computing (HPEC) Workshop (7th)., The original document contains color images.**		
14. ABSTRACT		
15. SUBJECT TERMS		

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**UU**	**28**	

and C_B describes a radix-4 decimation in time butterfly.

By comparing (4) with (1), the computational advantages of the manipulation leading to the FFT algorithm form (4) for base-4 designs are readily evident. In (4) the matrix products $C_{M1}X$ and $C_{M2}Y^t$ involve only exchanges of real and imaginary parts plus additions because the elements of C_{M1} and C_{M2} contain only ± 1 or $\pm j$, whereas the product CX in (1) requires complex multiplications. Also, the size of the coefficient matrix W_M in (4) is $(N/b)\times(N/b)$ vs. the $N \times N$ size of C in (1); consequently the number of overall direct multiplications in (4) is reduced by a factor of x16 compared to the direct form (1) on which past systolic FFT implementations are based. Note that distribution of the elements in C_{M1} and C_{M2} does not impose significant bandwidth requirements because full complex numbers are not used.

The overall processing for an M point FFT is done using the factorization $M=N_1N_2$, followed by a series of "row/column" 1-D FFTs on N_1 and N_2. Each of these 1-D FFTs is performed by a secondary factorization into 2-D FFTs according to (4) and again using a "row/column" approach. The first equation in (4) is equivalent to performing a "column" FFT and the second equation is equivalent to performing a "row" FFT. In between there is a "twiddle factor" multiplication by the W^p in W_M (3). Because both the row and column FFTs in the secondary factorization (4) are broken down entirely into sets of 4-point DFTs, they can be done without complex multiplications. Also, the usual matrix transpose in between column and row DFTs is not necessary.

A systolic array architecture mapping was performed using the mathematical formulation (4) as input to the mapping tool SPADE [2]. Behavioral simulations of this architecture using a register transfer level simulator verify its operation. Performance estimates of the FFT computation times are shown in Table 1 for a variety of transform sizes.

Size (points)	T (cycles/DFT)	T (μsec/DFT)	Multipliers	Adders
256	210	1.0	4	32
512	274	1.3	8	64
1024	658	3.1	8	64
2048	914	4.3	16	128
4096	2322	10.8	16	128
8192	3346	15.6	32	256

Table 1. Performance estimates and arithmetic requirements for various transform sizes (16-bits fixed point) based on a partially populated Altera Stratix EP1S60 "medium speed grade" FPGA chip. In this table "T" is the throughput. (Computational latency for each transform size above is approximately equal to the inverse of the throughput time/DFT).

Although Table 1 only shows transforms that are powers of two, the base-4 FFT lengths are not limited to powers of two or four. For example, the base-4 FFT is capable of 29 transform lengths from 256 to 65,536 *vs*. only 5 possible lengths for a radix-4 pipelined FFT.

The single biggest drawback to past use of systolic arrays has been the substantial arithmetic hardware that is normally required because systolic approaches use a number of complex multipliers equal to the size of the transform. Thus, a 1024-point DFT would require 1024 complex multipliers, compared to the 8 multipliers shown in Table 1 for the base-4 FFT.

Traditional "pipelined" FFTs, although computationally efficient, are difficult to map into VLSI because in general each butterfly, delay/commutator, and twiddle factor ROM has a different circuit design and/or its operation varies from stage to stage. Also, the butterflies do not usually work with 100% resource efficiency, the designs are limited to transform lengths that are powers of two or four, they are architecturally suited only for a 1-D DFT or 2-D DFT but not both, and it is difficult to build scalable designs because of their irregularity and large granularity. Finally, the latency (time to do the first DFT in a series) is low because the pipeline has to be "filled" first. Alternatively, the base-4 FFT architecture is comprised of simple, identical, small processing elements (PEs), arranged in regular arrays with each PE operating at near 100% efficiency. Performance figures in Table 1 compare very well to larger custom ASIC designs and recent FPGA implementations.

[1] C. C. W. Hui, T. J. Ding, J. V. McCanny, and R. F. Woods, "A New FFT Architecture and Chip Design for Motion Compensation based on Phase Correlation," Proc. Int. Conf. on Application Specific Systems, Architectures and Processors (ASAP 96), pp. 83-92.
[2] J. Greg Nash, "Hardware Efficient Base-4 Systolic Architecture for Computing the Discrete Fourier Transform," Proc. 2002 IEEE Workshop on Signal Processing Systems (SIPS'02), pp. 87-92.

High Performance Scalable Base-4 Fast Fourier Transform Mapping

Greg Nash
Centar

2003 High Performance Embedded Computing Workshop

www.centar.net

Outline

- Base-4 transformation for calculating DFT
- Mapping methodology
- Direct form DFT architecture
- FFT architecture
- Performance

Discreet Fourier Transform

- **Mathematical form:**

$$Z[k] = \sum_{n=1}^{N} X[n]\, e^{-I(2\pi/N)(k-1)(n-1)} \qquad k = 1, 2 \ldots N$$

- **Matrix form *Z=CX*:**
  (*N*=16)

$$Z = \begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & w & w^{2} & w^{3} & w^{4} & w^{5} & w^{6} & w^{7} & w^{8} & w^{9} & w^{10} & w^{11} & w^{12} & w^{13} & w^{14} & w^{15} \\
1 & w^{2} & w^{4} & w^{6} & w^{8} & w^{10} & w^{12} & w^{14} & 1 & w^{2} & w^{4} & w^{6} & w^{8} & w^{10} & w^{12} & w^{14} \\
1 & w^{3} & w^{6} & w^{9} & w^{12} & w^{15} & w^{2} & w^{5} & w^{8} & w^{11} & w^{14} & w & w^{4} & w^{7} & w^{10} & w^{13} \\
1 & w^{4} & w^{8} & w^{12} & 1 & w^{4} & w^{8} & w^{12} & 1 & w^{4} & w^{8} & w^{12} & 1 & w^{4} & w^{8} & w^{12} \\
1 & w^{5} & w^{10} & w^{15} & w^{4} & w^{9} & w^{14} & w^{3} & w^{8} & w^{13} & w^{2} & w^{7} & w^{12} & w & w^{6} & w^{11} \\
1 & w^{6} & w^{12} & w^{2} & w^{8} & w^{14} & w^{4} & w^{10} & 1 & w^{6} & w^{12} & w^{2} & w^{8} & w^{14} & w^{4} & w^{10} \\
1 & w^{7} & w^{14} & w^{5} & w^{12} & w^{3} & w^{10} & w & w^{8} & w^{15} & w^{6} & w^{13} & w^{4} & w^{11} & w^{2} & w^{9} \\
1 & w^{8} & 1 & w^{8} & 1 & w^{8} & 1 & w^{8} & 1 & w^{8} & 1 & w^{8} & 1 & w^{8} & 1 & w^{8} \\
1 & w^{9} & w^{2} & w^{11} & w^{4} & w^{13} & w^{6} & w^{15} & w^{8} & w & w^{10} & w^{3} & w^{12} & w^{5} & w^{14} & w^{7} \\
1 & w^{10} & w^{4} & w^{14} & w^{8} & w^{2} & w^{12} & w^{6} & 1 & w^{10} & w^{4} & w^{14} & w^{8} & w^{2} & w^{12} & w^{6} \\
1 & w^{11} & w^{6} & w & w^{12} & w^{7} & w^{2} & w^{13} & w^{8} & w^{3} & w^{14} & w^{9} & w^{4} & w^{15} & w^{10} & w^{5} \\
1 & w^{12} & w^{8} & w^{4} & 1 & w^{12} & w^{8} & w^{4} & 1 & w^{12} & w^{8} & w^{4} & 1 & w^{12} & w^{8} & w^{4} \\
1 & w^{13} & w^{10} & w^{7} & w^{4} & w & w^{14} & w^{11} & w^{8} & w^{5} & w^{2} & w^{15} & w^{12} & w^{9} & w^{6} & w^{3} \\
1 & w^{14} & w^{12} & w^{10} & w^{8} & w^{6} & w^{4} & w^{2} & 1 & w^{14} & w^{12} & w^{10} & w^{8} & w^{6} & w^{4} & w^{2} \\
1 & w^{15} & w^{14} & w^{13} & w^{12} & w^{11} & w^{10} & w^{9} & w^{8} & w^{7} & w^{6} & w^{5} & w^{4} & w^{3} & w^{2} & w
\end{bmatrix} X$$

$$\boldsymbol{W = e^{-2\pi I(n-1)(k-1)/N}}$$

- **Multiplications = *N* ^{2}**

Base-4 Matrix Equation

- Find reordering permutation P

$$X_{b=4} = P \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ \vdots \\ X_{N-3} \\ X_{N-2} \\ X_{N-1} \\ X_N \end{bmatrix} = \begin{bmatrix} X_1 \\ X_{1+N/4} \\ X_{1+N/2} \\ X_{1+3N/4} \\ X_2 \\ \vdots \\ X_{N/4} \\ X_{N/2} \\ X_{3N/4} \\ X_N \end{bmatrix}, \quad and \quad Z_{b=4} = P\ Z$$

- DFT matrix equation becomes

$$X_b = C_b Z_b$$

where

$$C_b = PCP^t$$

Base-4 Coefficient Matrix

$$C_b = \begin{bmatrix} d1\begin{bmatrix}1&1&1&1\\1&1&1&1\\1&1&1&1\\1&1&1&1\end{bmatrix} & d2\begin{bmatrix}1&1&1&1\\1&1&1&1\\1&1&1&1\\1&1&1&1\end{bmatrix} & d3\begin{bmatrix}1&1&1&1\\1&1&1&1\\1&1&1&1\\1&1&1&1\end{bmatrix} & d4\begin{bmatrix}1&1&1&1\\1&1&1&1\\1&1&1&1\\1&1&1&1\end{bmatrix} \\ d1\begin{bmatrix}1&-I&-1&I\\1&-I&-1&I\\1&-I&-1&I\\1&-I&-1&I\end{bmatrix} & W\,d2\begin{bmatrix}1&-I&-1&I\\1&-I&-1&I\\1&-I&-1&I\\1&-I&-1&I\end{bmatrix} & W^2\,d3\begin{bmatrix}1&-I&-1&I\\1&-I&-1&I\\1&-I&-1&I\\1&-I&-1&I\end{bmatrix} & W^3\,d4\begin{bmatrix}1&-I&-1&I\\1&-I&-1&I\\1&-I&-1&I\\1&-I&-1&I\end{bmatrix} \\ d1\begin{bmatrix}1&-1&1&-1\\1&-1&1&-1\\1&-1&1&-1\\1&-1&1&-1\end{bmatrix} & W^2\,d2\begin{bmatrix}1&-1&1&-1\\1&-1&1&-1\\1&-1&1&-1\\1&-1&1&-1\end{bmatrix} & W^4\,d3\begin{bmatrix}1&-1&1&-1\\1&-1&1&-1\\1&-1&1&-1\\1&-1&1&-1\end{bmatrix} & W^6\,d4\begin{bmatrix}1&-1&1&-1\\1&-1&1&-1\\1&-1&1&-1\\1&-1&1&-1\end{bmatrix} \\ d1\begin{bmatrix}1&I&-1&-I\\1&I&-1&-I\\1&I&-1&-I\\1&I&-1&-I\end{bmatrix} & W^3\,d2\begin{bmatrix}1&I&-1&-I\\1&I&-1&-I\\1&I&-1&-I\\1&I&-1&-I\end{bmatrix} & W^6\,d3\begin{bmatrix}1&I&-1&-I\\1&I&-1&-I\\1&I&-1&-I\\1&I&-1&-I\end{bmatrix} & W^9\,d4\begin{bmatrix}1&I&-1&-I\\1&I&-1&-I\\1&I&-1&-I\\1&I&-1&-I\end{bmatrix} \end{bmatrix}$$

$$d1=\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix};\quad d2=\begin{bmatrix}1&0&0&0\\0&-I&0&0\\0&0&-1&0\\0&0&0&I\end{bmatrix};\ d3=\begin{bmatrix}1&0&0&0\\0&-1&0&0\\0&0&1&0\\0&0&0&-1\end{bmatrix};\ d4=\begin{bmatrix}1&0&0&0\\0&-I&0&0\\0&0&-1&0\\0&0&0&-I\end{bmatrix}$$

Base-4 DFT Matrix Equation (Compact Form)

- **Form for *N*=16**

$$Y = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & W & W^2 & W^3 \\ 1 & W^2 & W^4 & W^6 \\ 1 & W^3 & W^6 & W^9 \end{bmatrix} \bullet \left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -I & 1 & I \\ 1 & -1 & 1 & -1 \\ 1 & I & 1 & -I \end{bmatrix} \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \\ x_5 & x_6 & x_7 & x_8 \\ x_9 & x_{10} & x_{11} & x_{12} \\ x_{13} & x_{14} & x_{15} & x_{16} \end{bmatrix} \right)$$

$$\begin{bmatrix} z_1 & z_2 & z_3 & z_4 \\ z_5 & z_6 & z_7 & z_8 \\ z_9 & z_{10} & z_{11} & z_{12} \\ z_{13} & z_{14} & z_{15} & z_{16} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -I & 1 & I \\ 1 & -1 & 1 & -1 \\ 1 & I & 1 & -I \end{bmatrix} Y^t$$

"•"= element by element multiply

- **General Form**

$$Y = W_M^t \cdot C_{M1} X_b$$

$$Z_b = C_{M2} Y^t$$

$$Z_b = \begin{bmatrix} Z_1 & & Z_{N/4} \\ Z_{1+N/4} & \cdots & Z_{N/2} \\ Z_{1+N/2} & & Z_{3N/4} \\ Z_{1+3N/4} & & Z_N \end{bmatrix}, \quad X_b = \begin{bmatrix} X_1 & & X_{N/4} \\ X_{1+N/4} & \cdots & X_{N/2} \\ X_{1+N/2} & & X_{3N/4} \\ X_{1+3N/4} & & X_N \end{bmatrix}$$

Base-4 DFT Equation Characteristics

- Coefficient matrices represent series of 4-point transforms:

$$C_{M1} = \left[C_B^t \mid C_B^t \mid \dots \right]^t$$

$$C_{M2} = \left[C_B \mid C_B \mid \dots \right] \quad \text{where} \quad C_B = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -I & -1 & I \\ 1 & -1 & 1 & -1 \\ 1 & I & -1 & I \end{bmatrix}$$

  $\Rightarrow$ Takes advantage of reduced arithmetic with radix $r = 4$ butterfly, but transform length not limited to $N = r^m$

  $\Rightarrow$ Transform length must be divisible by 16

- C_{M1} and C_{M2} contain only elements from the set $\{1, -1, -I, I\}$

  $\Rightarrow$ $C_{M1}X$ and $C_{M2}Y^t$ only involve complex additions

- Twiddle factor matrix W_M is of size $N/4 \times N/4$ rather than $N \times N$

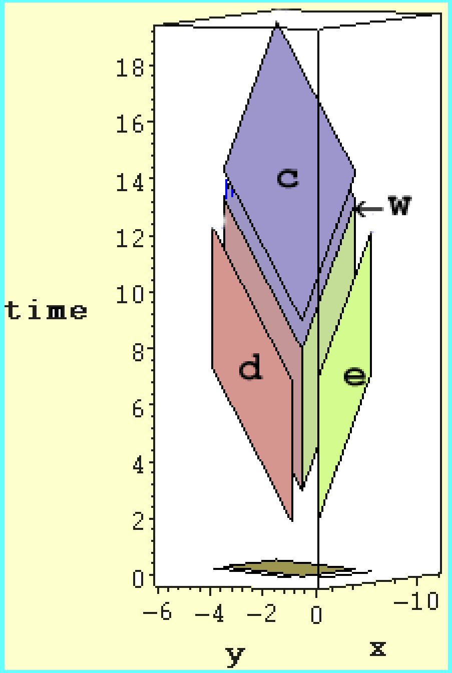
  $\Rightarrow$ *x16* fewer multiplies than original DFT equation ($Z=CX$)

Systolic Array Example: Matrix Multiply

- **Algorithm:**

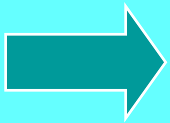
$$c[i,j]=\sum_{k=1}^{N} d[i,k]*e[k,j] \quad for \quad 1 \le i,j,k \le N$$

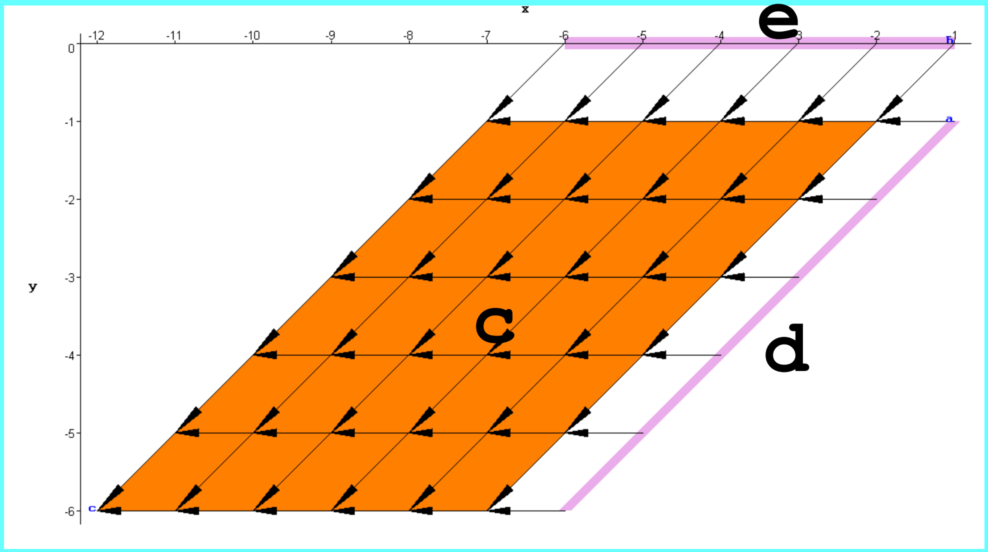
- **Space-time mapping: computations at {i,j,k} "mapped" to indices {time,x,y}**



"Space-Time" View

Project along time axis





Systolic Array: Each intersection point corresponds to a "processing element" (PE) that receives data from its neighbors, does a multiply-add, and passes the result to adjacent PEs, once per time cycle.

Find Systolic Architecture Using SPADE†

Mathematical Algorithm

$$Y = W_M^t \cdot C_{M1} X$$
$$Z = C_{M2} Y^t$$

Input Code

```
for j to N/4 do
 for k to N/4 do
   Y[j,k]:=WM[j,k]*add(CM1[j,i]*X[i,k],i=1..4);
 od;
 for k to 4 do
   Z[k,j] := add(CM2[k,i]*Y[j,i],i=1..N/4);
 od
od;
```

Architectural Constraints Objective Functions

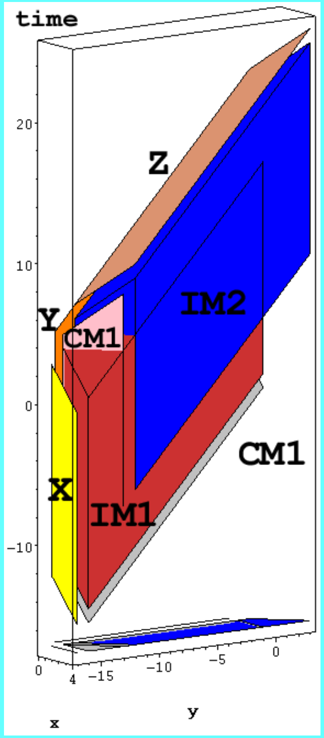
Variable position,
area, regularity, bandwidth

Automatic Search for Space-Time Transformations, T

$$\begin{bmatrix} time \\ x \\ y \end{bmatrix}_v = T_v \begin{bmatrix} i \\ j \\ k \end{bmatrix}$$

$$v \in \{X, Y, Z, CM1, CM2, WM\}$$

Simulator, Graphical Outputs



†Symbolic Parallel Algorithm Development Environment

SPADE Functionality

- SPADE accepts input statements of the affine form

$$x(A_x I + a_x) \quad depends \quad on \quad y(B_y I + b_y) \qquad for\ all\ I \in V(I)$$

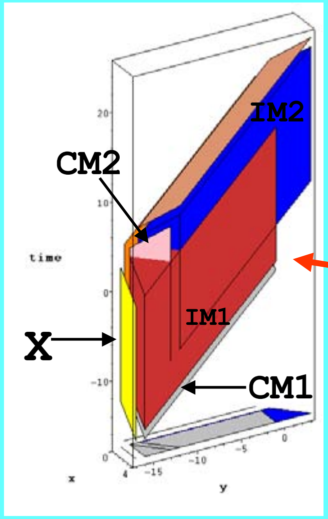
$$e.g.,\ x(2i, j+1) \equiv x\left(\begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} i \\ j \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix}\right)$$

  - **Where A_x,B_y/a_x,b_y are integer matrices/vectors, *S* is the dimension of the algorithm space and the "depends on" includes commutative and associative operators: *min, max, Σ, Π***

- SPADE finds latency optimal systolic designs subject to constraints imposed by scheduling, localization, reindexing, and allocation

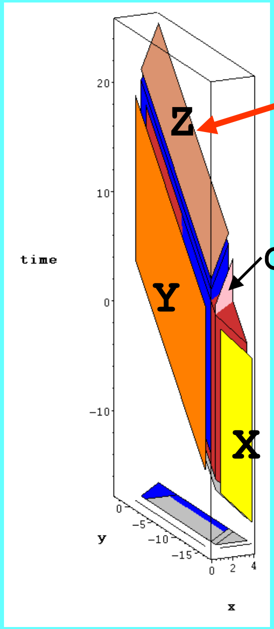
- Secondary objective functions used to select architectures are minimum area, maximum regularity and minimum network bandwidth

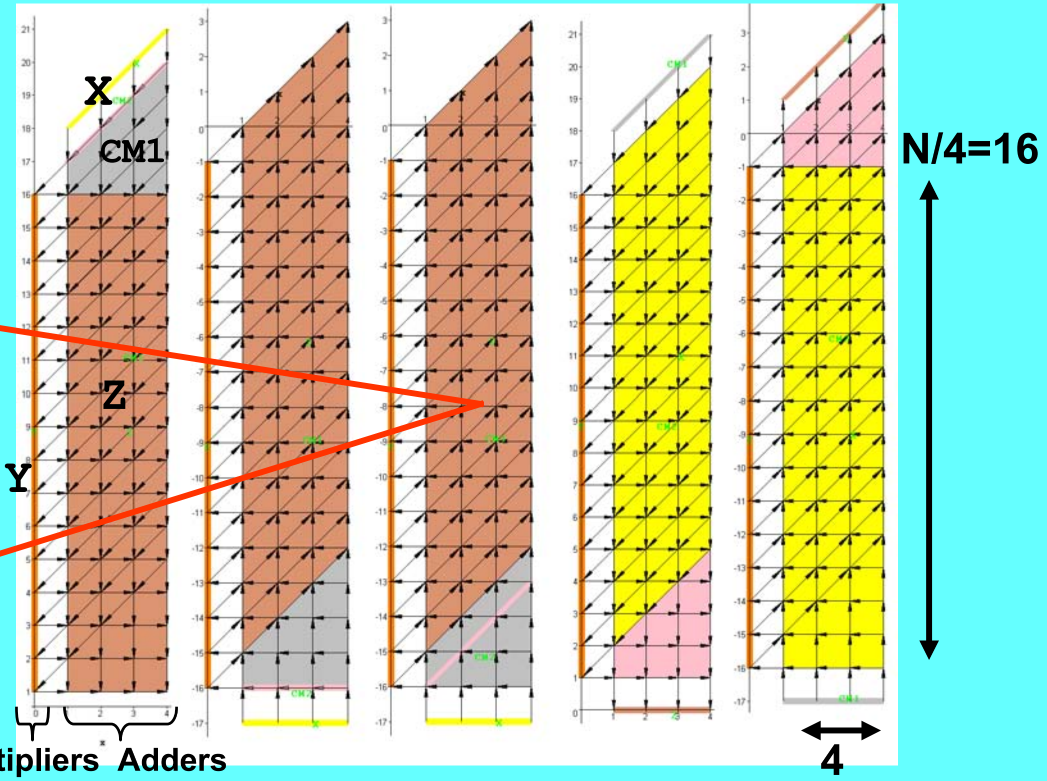
Systolic Array Designs: Minimum Area



$$Y = W_M^t \cdot C_{M1} X_b$$
$$Z_b = C_{M2} Y^t$$

Space-Time Views (N=64)





Example Systolic Array Views (*N*=64)

- Latency (cycles) = *N*/2 + 8
- Six unique designs
- Throughput (cycles/block) = *N*/4 + 6
- W_M mapped to same space-time location as *Y*
- *IM1* and *IM2* variables (SPADE created) perform matrix multiply/adds

Systolic Array Designs: Maximum Regularity

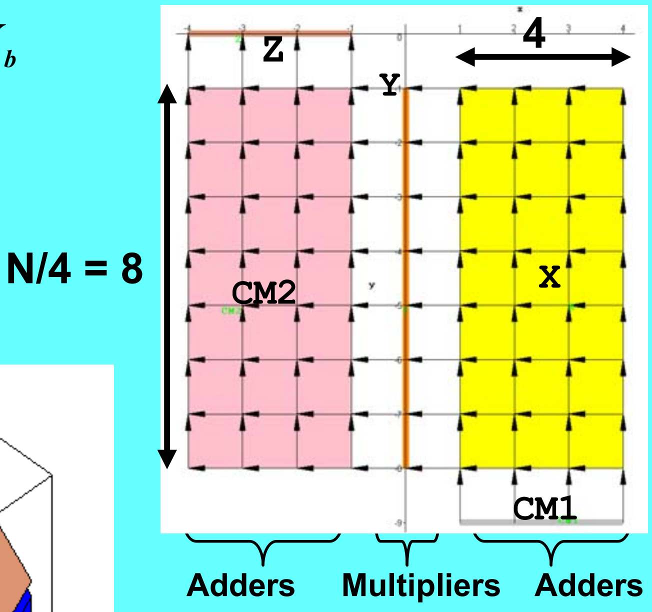
- Two unique designs found
- Throughput and latency optimal
- Latency (cycles) = *N*/2 + 8
- Throughput (cycles/block) = *N*/4 +1
- W_M mapped to same space-time position as *Y*

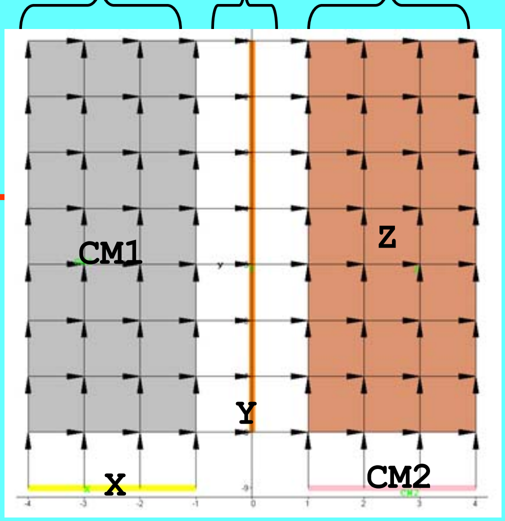
$$Y = W_M^t \cdot C_{M1} X_b$$
$$Z_b = C_{M2} Y^t$$

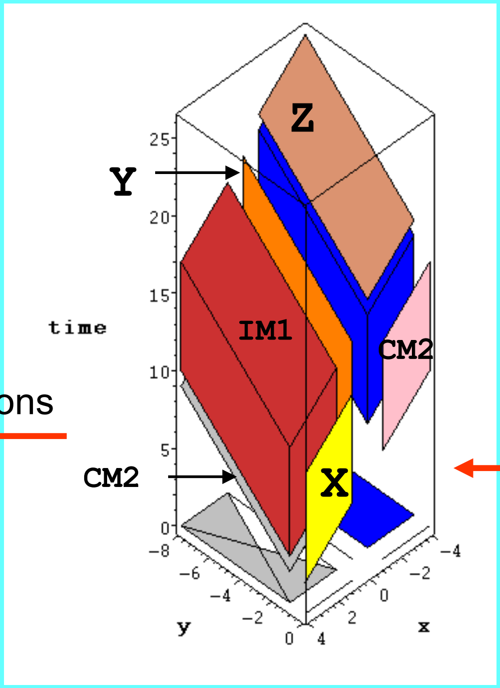
variable	*T*	*t*
Y	$\begin{bmatrix} 1 & 1 \\ 0 & 0 \\ -1 & 0 \end{bmatrix}$	[5 0 0]
IM1	$\begin{bmatrix} 1 & 1 & 1 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix}$	[0 5 0]
CM1	$\begin{bmatrix} 1 & 1 \\ 0 & -1 \\ -1 & 0 \end{bmatrix}$	[0 5 0]
X	$\begin{bmatrix} 1 & 1 \\ -1 & 0 \\ 0 & 0 \end{bmatrix}$	[0 5 0]
Z	$\begin{bmatrix} -1 & 1 \\ 1 & 0 \\ 0 & -1 \end{bmatrix}$	[19 -5 0]
IM2	$\begin{bmatrix} 1 & 1 & -1 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix}$	[10 -5 0]
CM2	$\begin{bmatrix} -1 & 1 \\ 1 & 0 \\ 0 & 0 \end{bmatrix}$	[10 –5 0]

Transformations

Systolic Arrays (N=32)



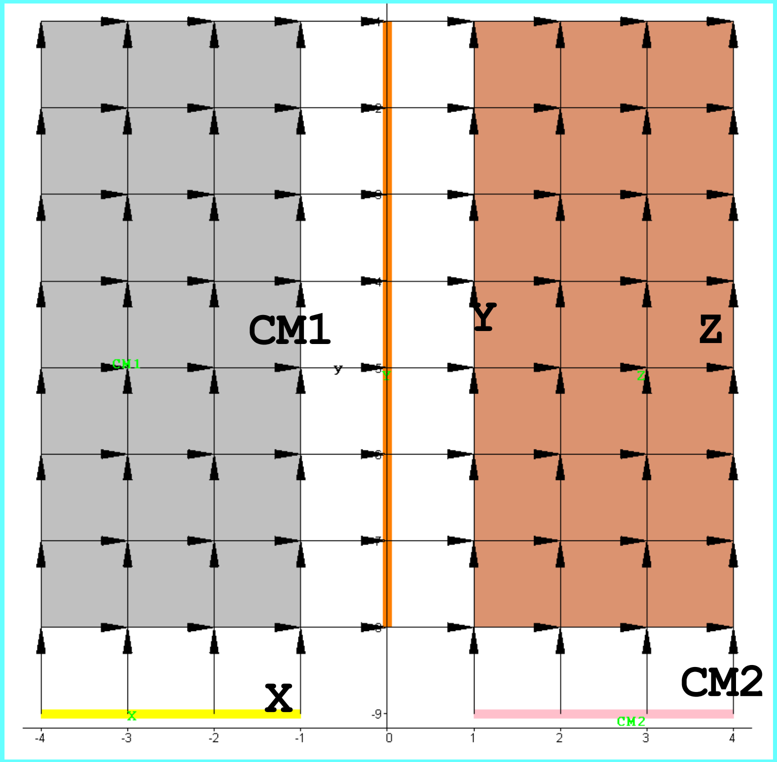




Space-time view (N=32)

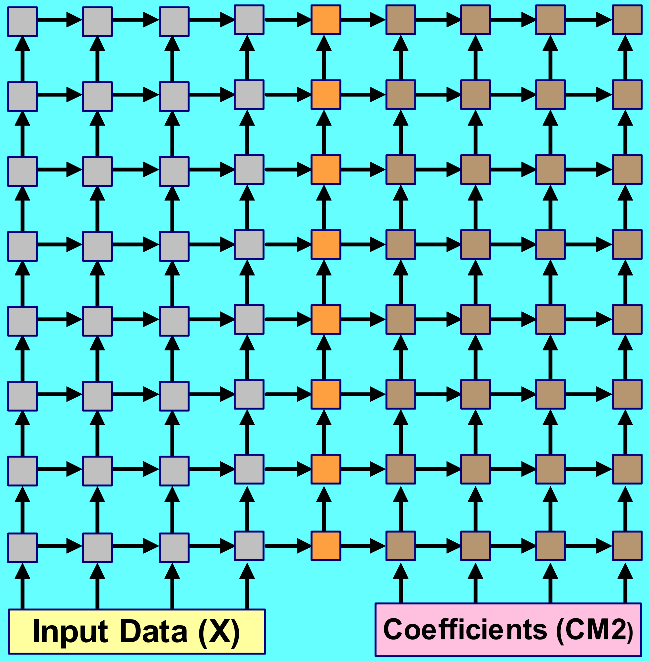
Systolic Architecture to Array Design

Systolic Architecture (N=32)



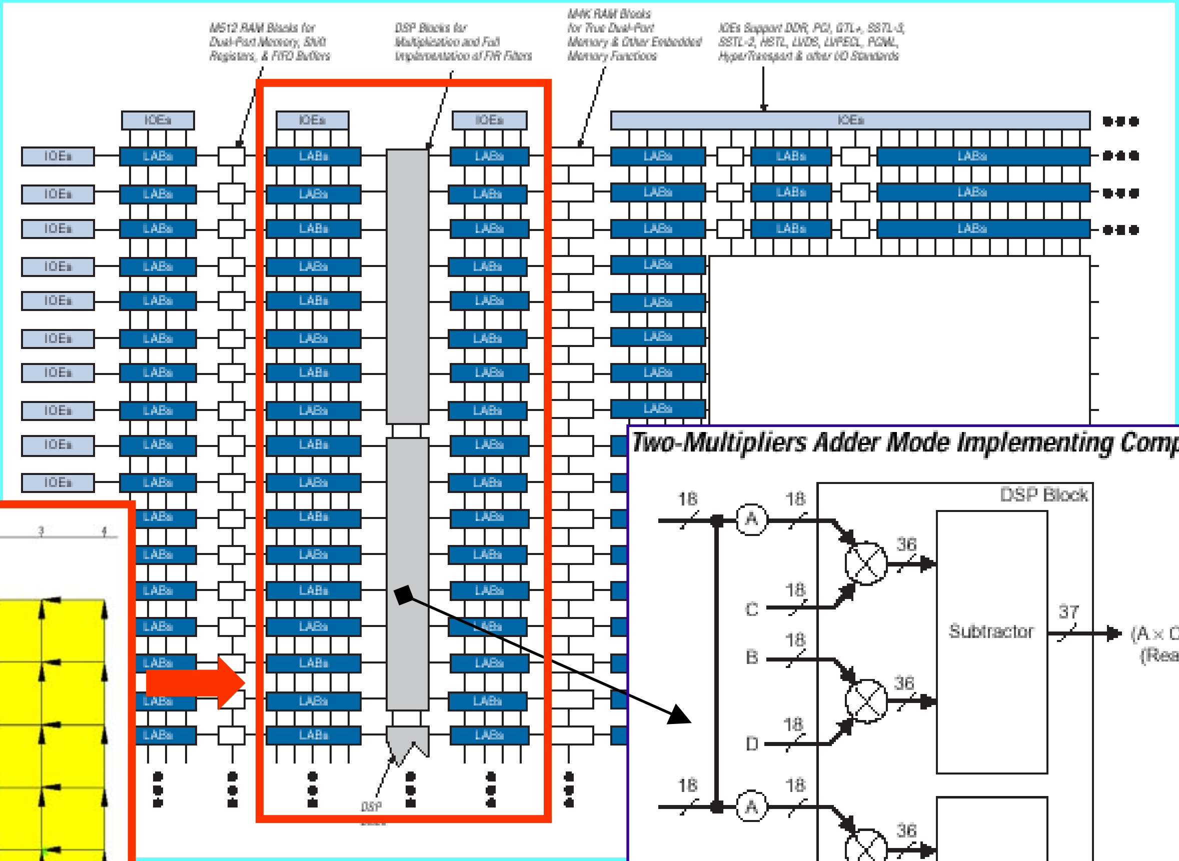
$\Rightarrow$

Array Design (N=32)

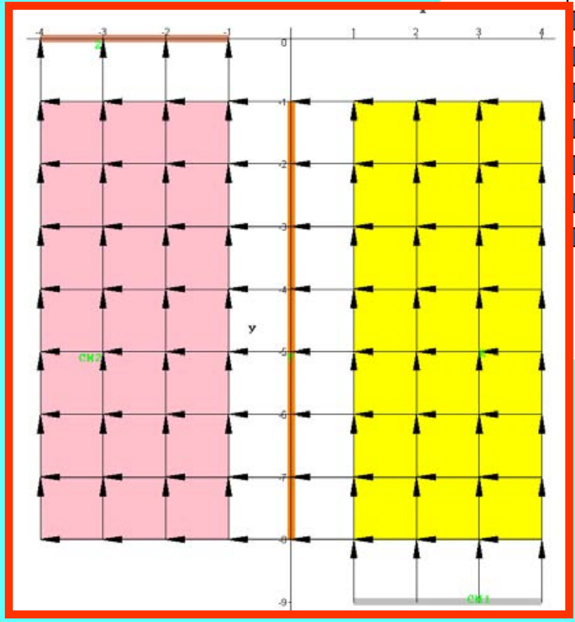


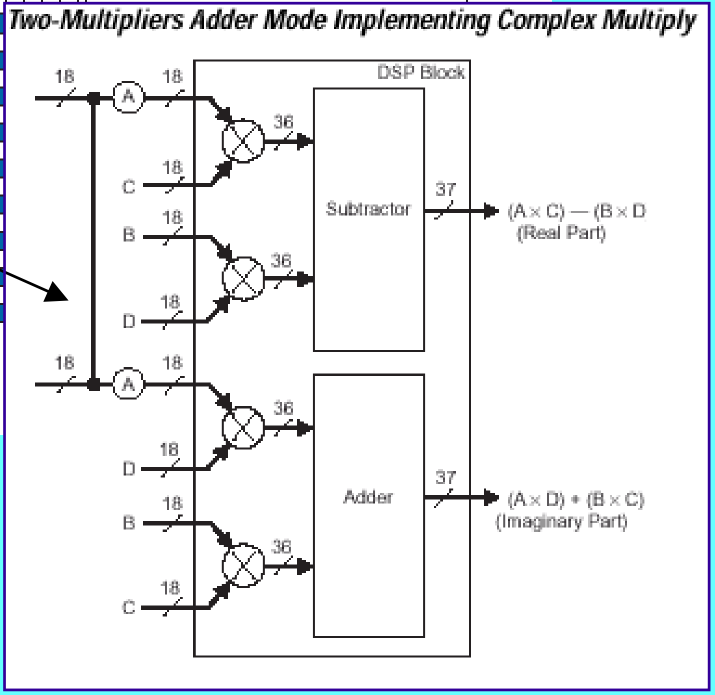
🟫	Processing Element 1: 2 registers, 1 adder
🟧	Multiplier
⬜	Processing Element 2: 2 registers, 1 adder
↑→	Data flow bus

Altera Stratix FPGA: DFT Mapping


M512 RAM Blocks for Dual-Port Memory, Shift Registers, & FIFO Buffers
DSP Blocks for Multiplication and Full Implementation of FIR Filters
M4K RAM Blocks for True Dual-Port Memory & Other Embedded Memory Functions
IOEs Support DDR, PCI, GTL+, SSTL-3, SSTL-2, HSTL, LVDS, LVPECL, PCML, HyperTransport & other I/O Standards
IOEs
LABs
DSP

Systolic DFT Array





1D FFT via Factorization

- Factor $N = N_1 * N_2$
- Creat a 2-D matrix with N_1 rows by N_2 columns, (assume $N_1 > N_2$),
- Do N_2 1-D "column" DFTs followed by N_1 "row" DFTs:

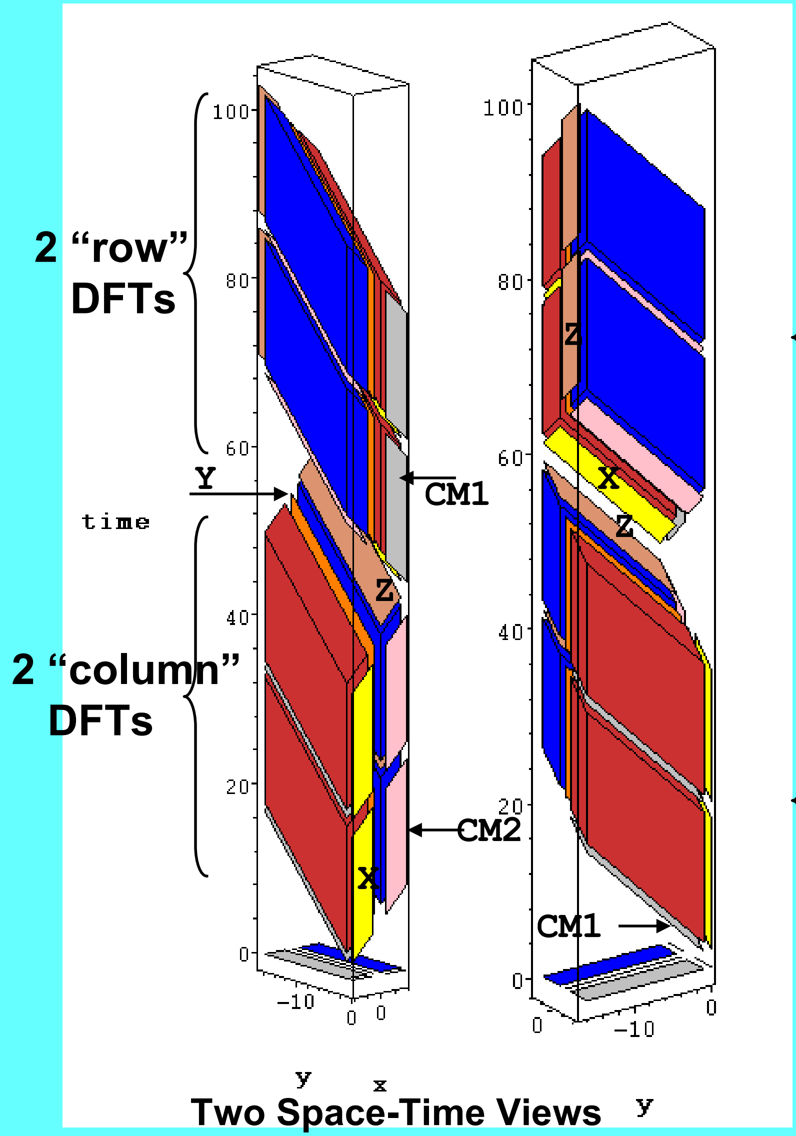
$$Y = W_{N1} * X$$

$$Y' = W_{N} \bullet Y$$

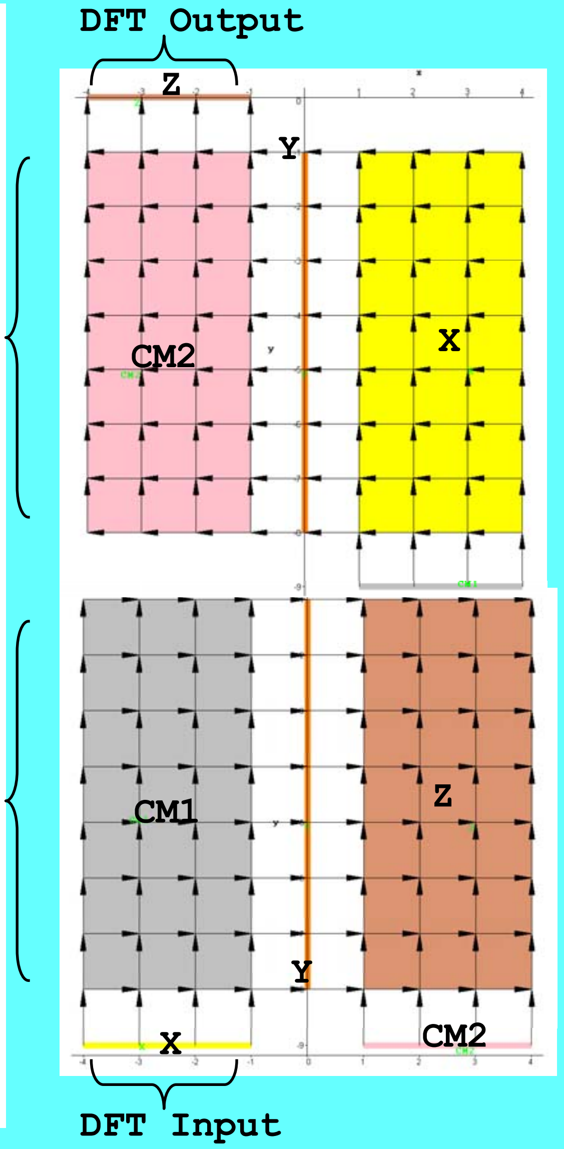
$$Z = Y' * W_{N2}$$

- If $N_1 \approx N_2$ then (linear) array size can be reduced from O($N_1 \; N_2$) to O(N_1) with minimal effect on throughput:
  - Cycles for $N/4$ array (no factorization) = $N/4 + 1$
  - *Cycles for* $N_1/4$ array = $N_1 (N_1/4 + 1) + N_1 (N_1/4 + 1)$ + twiddle mult $\approx N/2$
- Can do 2-D DFT by not performing twiddle multiplication W_N
- Use base-4 DFT mapping to do all row/column DFTs

Base-4 Factorization Architecture

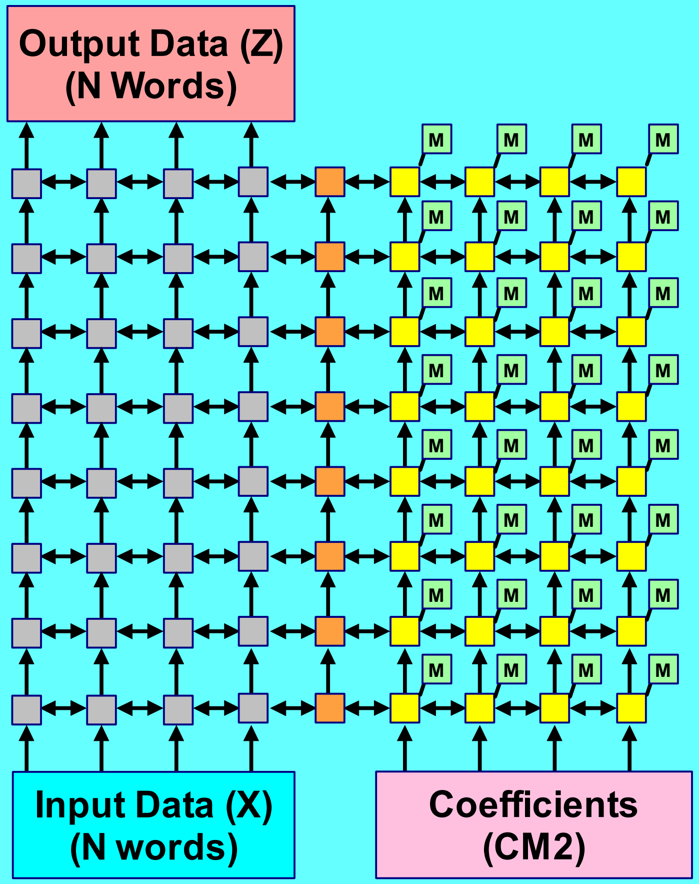


Two Space-Time Views
(only two of N_1 iterations shown)



- N = 1024 points
- $N = N_1 * N_2$
- $N_1 = N_2 = 32$
- Uses both of the two optimal systolic designs
- Twiddle multiplications not shown
- Throughput/latency optimal except for interstage delay

Two DFT Architectures Combined



- Shown for N = 1024 points
- $N = N_1 * N_2$
- $N_1 = N_2 = 32$
- M = 512 bits (16 bit word)

🟨 Processing Element 1: 2 registers, 1 adder

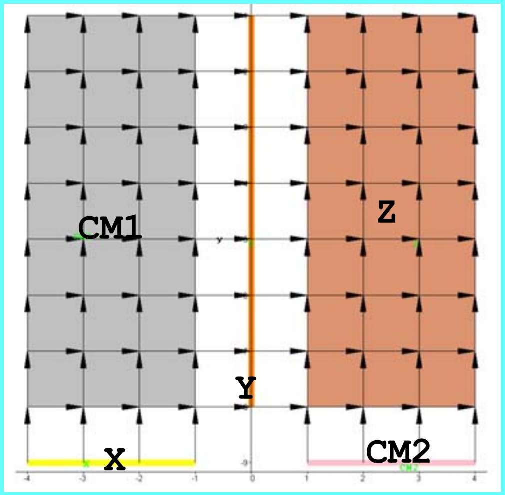
🟩 Memory

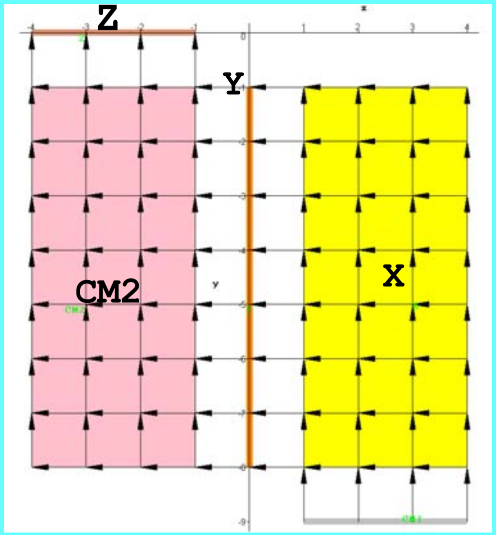
🟧 Multiplier

⬜ Processing Element 2: 2 registers, 1 adder

↕↔ Local data flow bus

1st to 2nd Stage Data Formatting Problem (32 Point DFT)





- DFT data positions of 1st stage output sequences

$$\begin{bmatrix}1&9&17&25\\2&10&18&26\\3&11&19&27\\4&12&20&28\\5&13&21&29\\6&14&22&30\\7&15&23&31\\8&16&24&32\end{bmatrix}\begin{bmatrix}1&9&17&25\\2&10&18&26\\3&11&19&27\\4&12&20&28\\5&13&21&29\\6&14&22&30\\7&15&23&31\\8&16&24&32\end{bmatrix}\;\cdots\cdots\;\begin{bmatrix}1&9&17&25\\2&10&18&26\\3&11&19&27\\4&12&20&28\\5&13&21&29\\6&14&22&30\\7&15&23&31\\8&16&24&32\end{bmatrix}$$

- Desired data positions for input sequences to 2nd stage

$$\begin{bmatrix}1&1&1&1\\1&1&1&1\\1&1&1&1\\1&1&1&1\\1&1&1&1\\1&1&1&1\\1&1&1&1\\1&1&1&1\end{bmatrix}\begin{bmatrix}2&2&2&2\\2&2&2&2\\2&2&2&2\\2&2&2&2\\2&2&2&2\\2&2&2&2\\2&2&2&2\\2&2&2&2\end{bmatrix}\;\cdots\cdots\;\begin{bmatrix}32&32&32&32\\32&32&32&32\\32&32&32&32\\32&32&32&32\\32&32&32&32\\32&32&32&32\\32&32&32&32\\32&32&32&32\end{bmatrix}$$

Interstage Data Formatting via "On-the-Fly" Permutations

- **New code with matrix rotation steps**

```
for n to N
  for  j to N/4 do
    for k to N/4 do
         Y[j,k] := WM[j,k]*add(CM1[j,i]*X[i,k],i=1..b) od;
       for k to b do
          Z[k,j] := add(CM2[k,i]*Y[j,i],i=1..N/4) od;
          WM := matrix_rotate(WM,"up");
          CM1 := matrix_rotate(CM1,"down");
          if n mod(b)=0 then CM2 := matrix_rotate(CM2,"down") fi;
  od;
od;
```

- **New DFT first stage output sequences**

$$\begin{bmatrix} 1 & 9 & 17 & 25 \\ 2 & 10 & 18 & 26 \\ 3 & 11 & 19 & 27 \\ 4 & 12 & 20 & 28 \\ 5 & 13 & 21 & 29 \\ 6 & 14 & 22 & 30 \\ 7 & 15 & 23 & 31 \\ 8 & 16 & 24 & 32 \end{bmatrix} \quad \begin{bmatrix} 2 & 10 & 18 & 26 \\ 3 & 11 & 19 & 27 \\ 4 & 12 & 20 & 28 \\ 5 & 13 & 21 & 29 \\ 6 & 14 & 22 & 30 \\ 7 & 15 & 23 & 31 \\ 8 & 16 & 24 & 32 \\ 1 & 9 & 17 & 25 \end{bmatrix} \quad \begin{bmatrix} 3 & 11 & 19 & 27 \\ 4 & 12 & 20 & 28 \\ 5 & 13 & 21 & 29 \\ 6 & 14 & 22 & 30 \\ 7 & 15 & 23 & 31 \\ 8 & 16 & 24 & 32 \\ 1 & 9 & 17 & 25 \\ 2 & 10 & 18 & 26 \end{bmatrix} \quad \ldots\ldots \quad \begin{bmatrix} 25 & 1 & 9 & 17 \\ 26 & 2 & 10 & 18 \\ 27 & 3 & 11 & 19 \\ 28 & 4 & 12 & 20 \\ 29 & 5 & 13 & 21 \\ 30 & 6 & 14 & 22 \\ 31 & 7 & 15 & 23 \\ 32 & 8 & 16 & 24 \end{bmatrix}$$

1-D DFT Performance Estimates

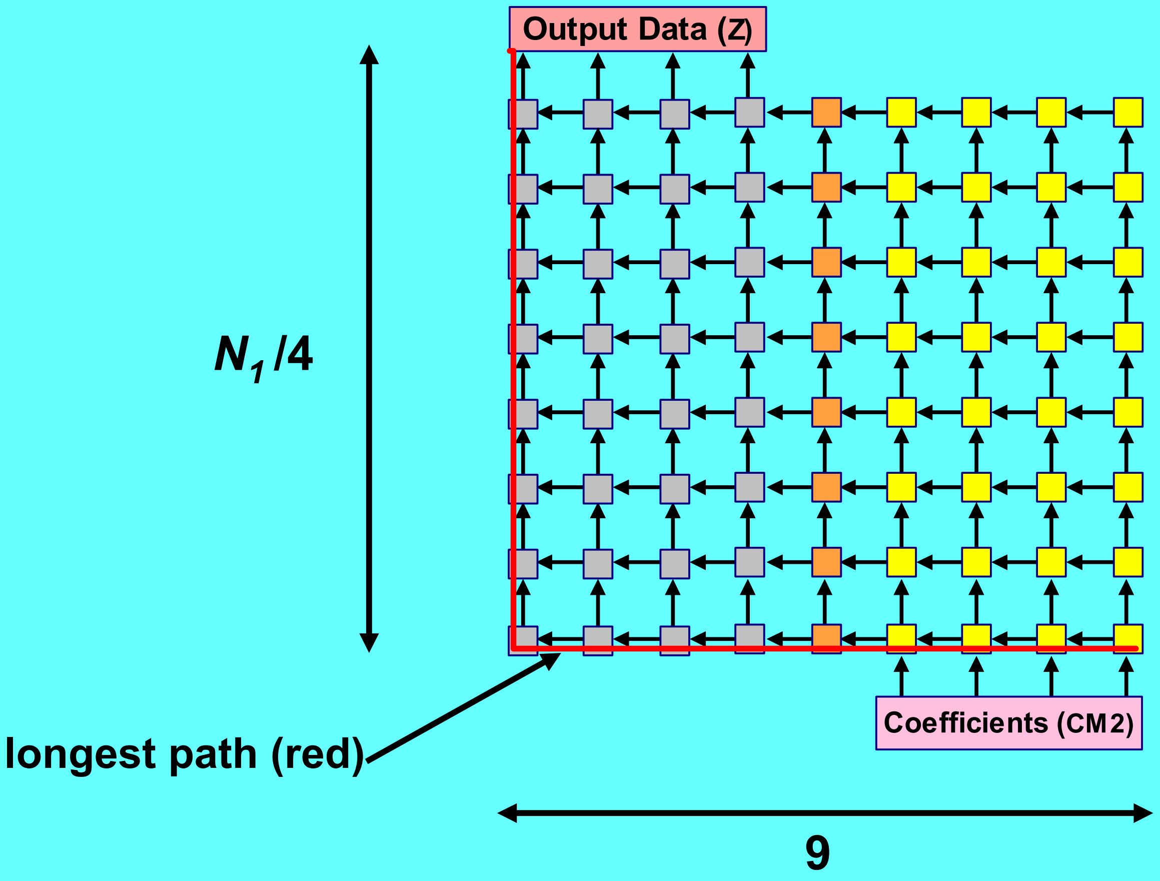
FFT Size	Throughput (cycles/DFT)	Throughput (μsec/DFT)	Multipliers	Adders
256	210	1.0	4	32
512	274	1.3	8	64
1024	671	3.1	8	64
2048	914	4.3	16	128
4096	2322	10.8	16	128
8192	3346	15.6	32	256

Based on:

- Register transfer level behavioral simulation of 1024 point DFT
- Partially populated layout
- Timing analysis using Altera Stratix EP1S60 FPGA chip
- 16 bit fixed-point word length

Latency

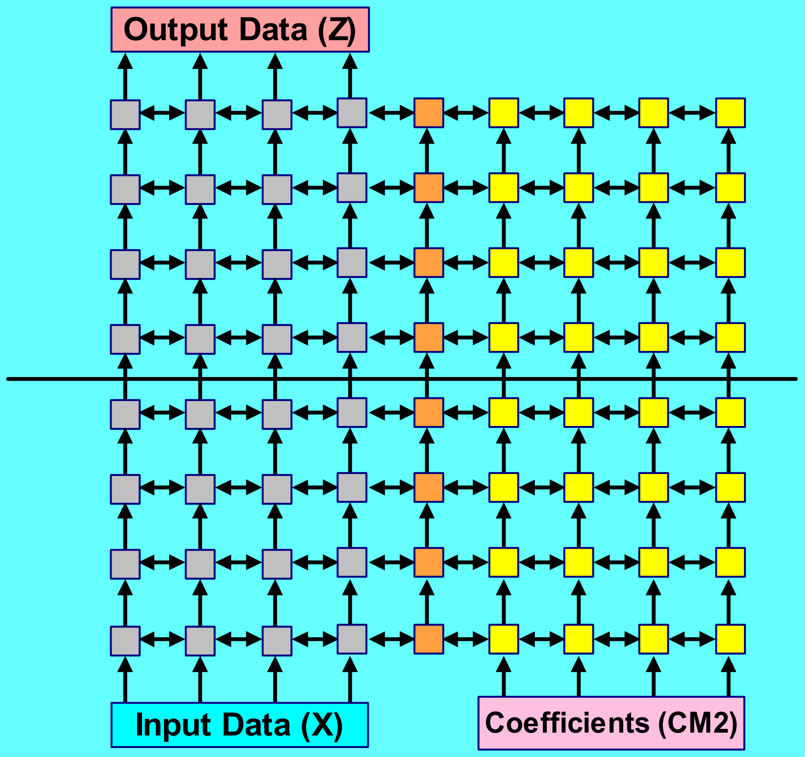
- **Base-4 FFT pipeline depth is nominally $N_1/4 + 9 \ll N$**



- **Latency (cycles) $\cong$ 1/Throughput (cycles^{-1}) when complete X available**

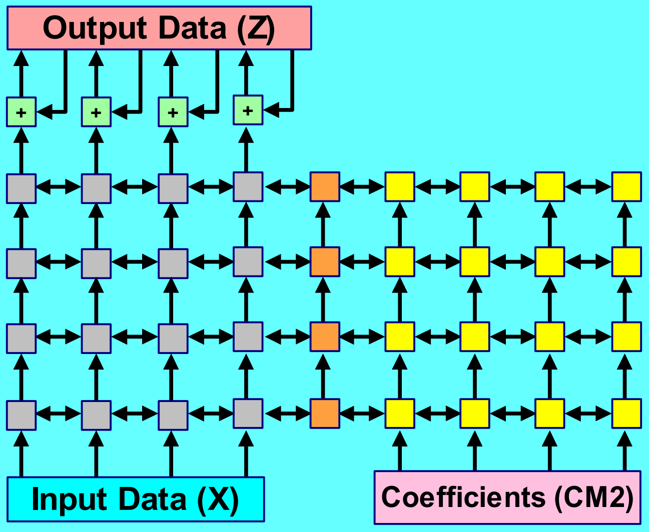
Partitioning to Scale Computations to Application

- Use an array "section" to perform partially processed result
- Partial results accumulated at output
- Memory needed scales with partition size



Fully Parallel Array

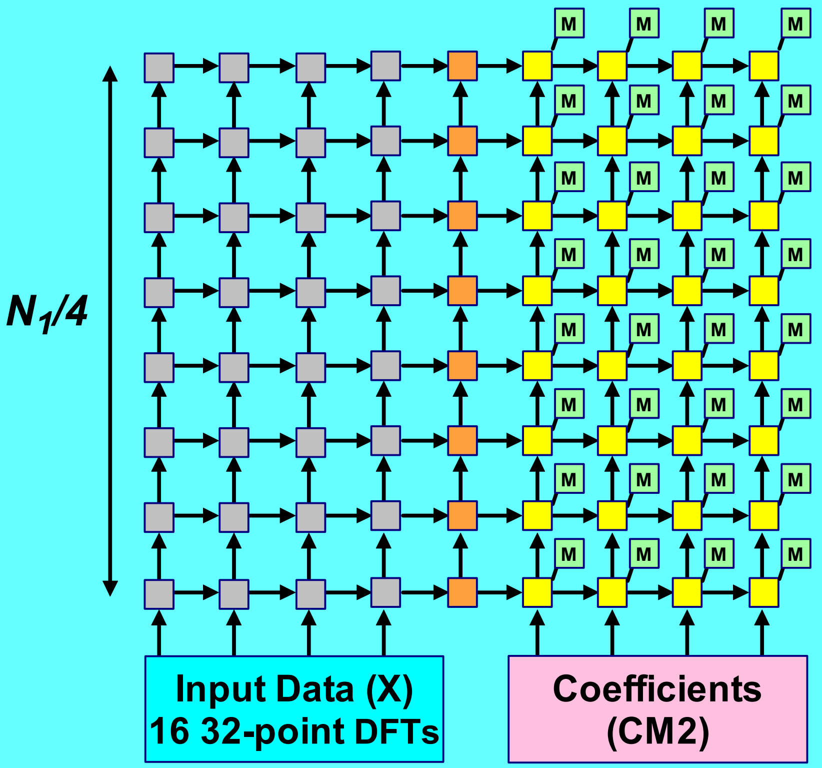
$\Rightarrow$



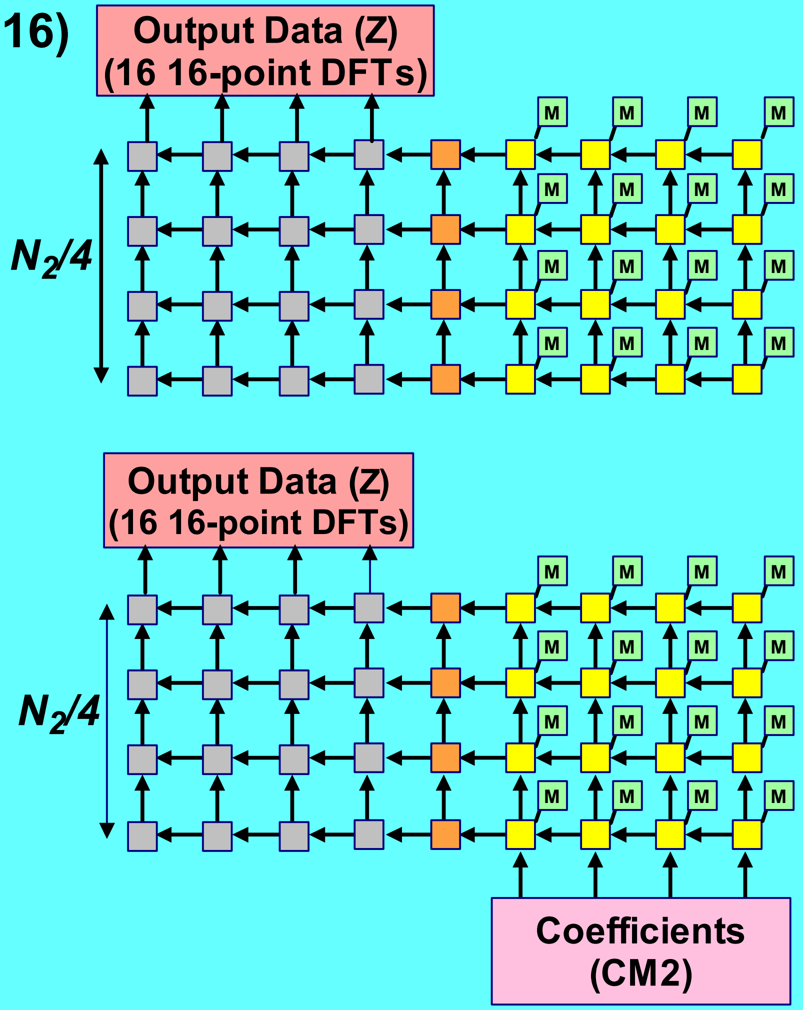
Partitioned Array

Non-Square 2-D Inputs ($N_1 \neq N_2$)

- Example: 512-point FFT (N_1 = 32, N_2 = 16)
- On-the-fly permutations for correct data placement



Columns: Compute 16 32-point DFTs



Output Data (Z) (16 16-point DFTs)
M
$N_2/4$
Output Data (Z) (16 16-point DFTs)
$N_2/4$
Coefficients (CM2)

Rows: Compute 2 sets of 16 16-point DFTs

Example Resource Usage[†]: 1024 Point DFT

Resource	Logic Cells	Flip flops	M512	M4K	DSP Blocks	Global Clocks
Usage	14717	9200	64	32	8	1
Percent Resources	26	15	11	11	44	17

[†] Altera Stratix EP1S60F1508C6 FPGA chip (16 bit fixed point)

Base-4 DFT Architecture Summary

- High performance 1-D and 2-D DFTs
- Based on latency and throughput optimal parallel circuits
- Transform size not restricted to $N = r^m$
- Latency $\approx$1/throughput when entire input block available
- Architecture is scaleable and easily parameterized
- Design is simple, regular, local and synchronous
- Fast convolutions naturally supported
- Natural partitioning strategies exist
- Pseudo-linear architecture good fit to latest generation of FPGA chips

Abstract Presentation Back to Agenda Next Abstract

More Information at www.centar.net

- "Automatic Generation of Systolic Array Designs For Reconfigurable Computing" , Proc. Engineering of Reconfigurable Systems and Algorithms (ERSA '02), International Multiconference in Computer Science, Las Vegas, Nevada, June 24, 2002.
  - General description of SPADE
  - Faddeev algorithm (Find *CX*+*D*, given *AX*=*B*, *X* is unknown)

- Constraint Directed CAD Tool For Automatic Latency-Optimal Implementations, SPIE ITCom 2002, Boston, Massachussetts, July 29-August 2, 2002.
  - Use of constraints as a filter of systolic designs
  - 2-D Discreet Fourier transforms using base-4 architecture